CS 110
Computer Architecture

VM, ES, & FPGA

Instructor:

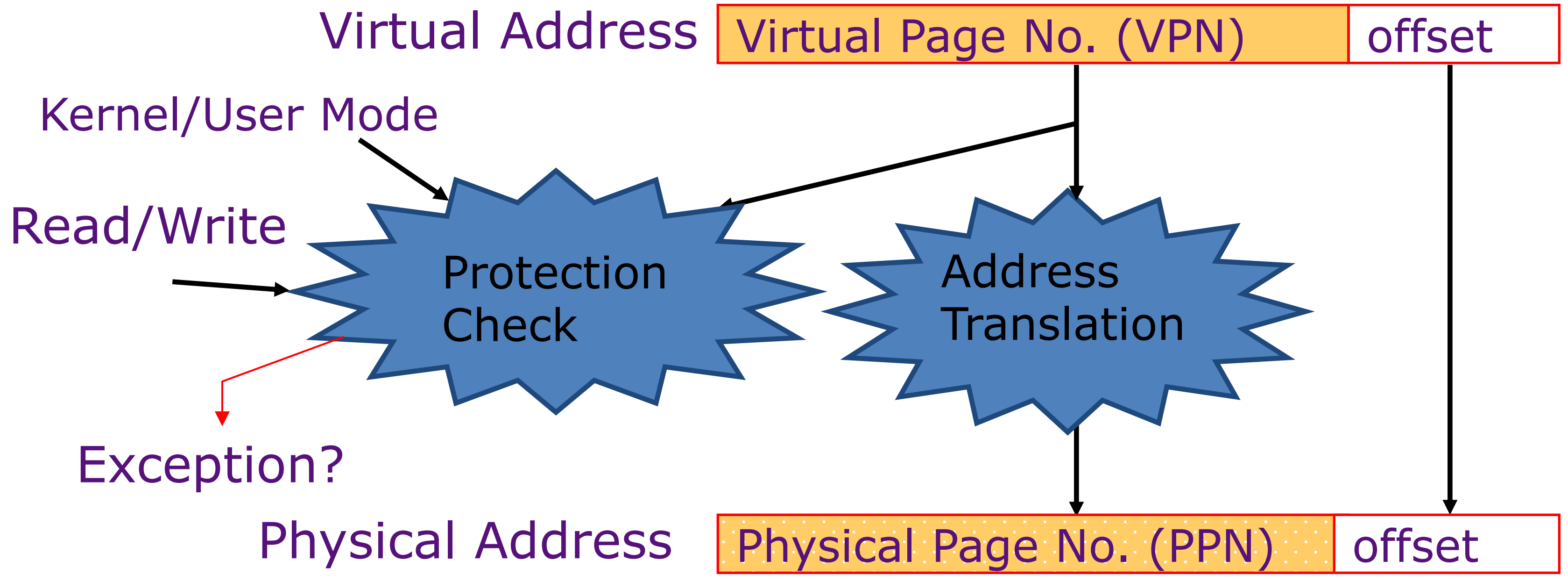
Sören Schwertfeger and Chundong Wang

https://robotics.shanghaitech.edu.cn/courses/ca/22s

School of Information Science and Technology SIST

ShanghaiTech University

Address Translation & Protection



- Every instruction and data access needs address translation and protection checks

- *A good VM design needs to be fast (~ one cycle) and space efficient*

Translation Lookaside Buffers (TLB)

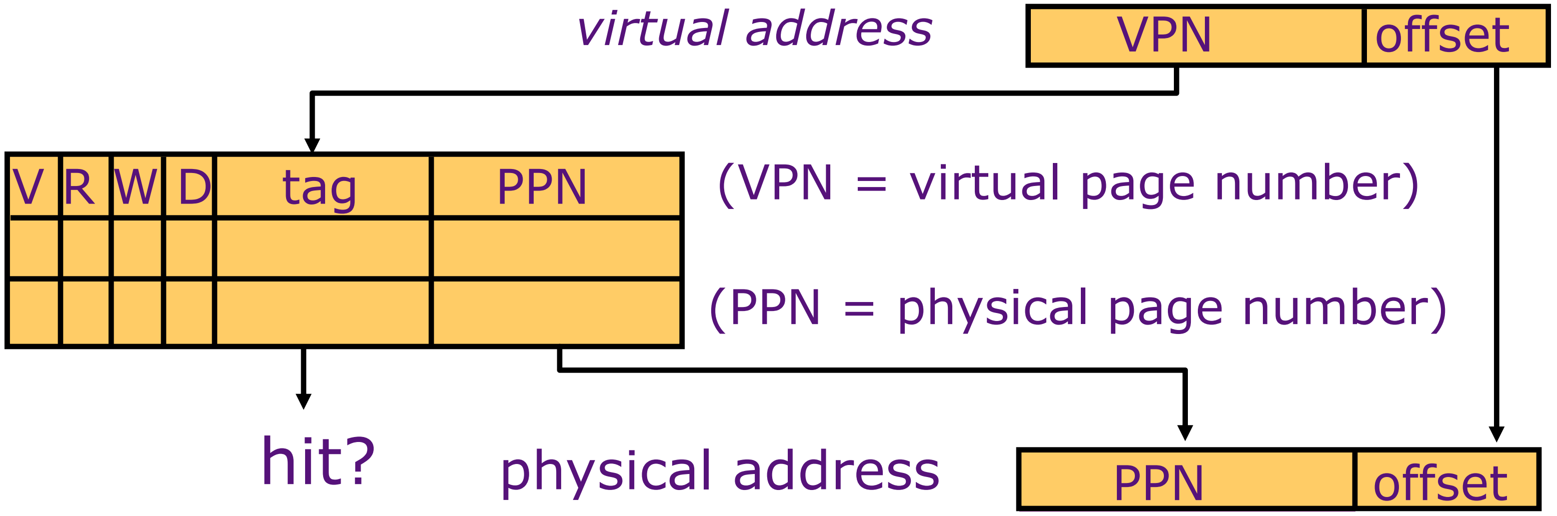
Address translation is very expensive!

In a two-level page table, each reference becomes several memory accesses

Solution: *Cache some translations in TLB*

TLB hit => *Single-Cycle Translation*

TLB miss => *Page-Table Walk to refill*



TLB Designs

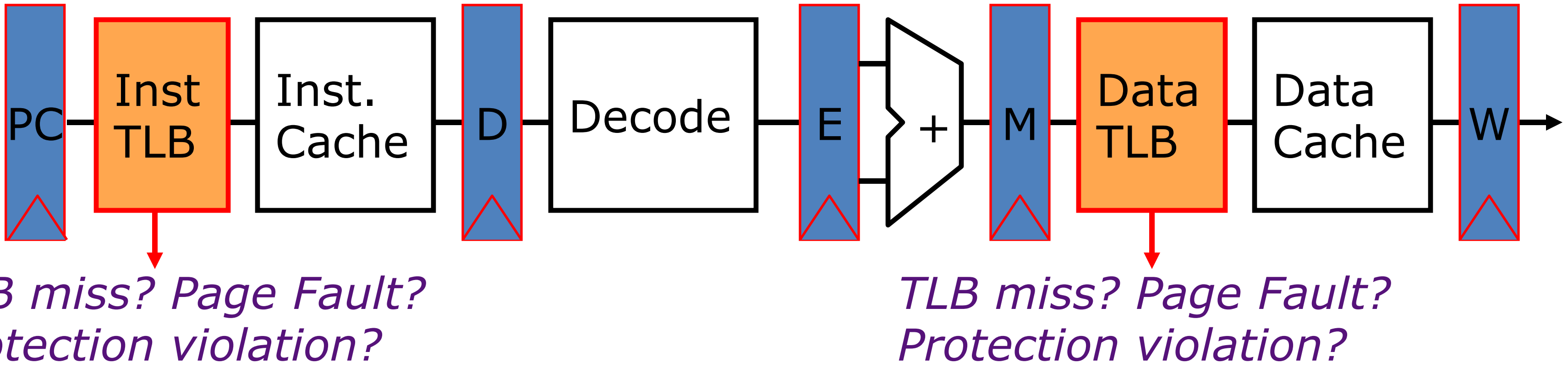
- Typically 32-128 entries, usually fully associative
  - Each entry maps a large page, hence less spatial locality across pages => more likely that two entries conflict
  - Sometimes larger TLBs (256-512 entries) are 4-8 way set-associative
  - Larger systems sometimes have multi-level (L1 and L2) TLBs
- Random or FIFO replacement policy
- Upon context switch? New VM space! Flush TLB …
- “TLB Reach”: Size of largest virtual address space that can be simultaneously mapped by TLB

Quiz

TLB Reach

- Given a TLB of 256 entries and the page offset in a page table entry (TLB) has 20 bits, what is the TLB reach?
  A. 64MB
  B. 128MB
  C. 256MB
  D. 512MB
  E. 1GB
  F. None of these

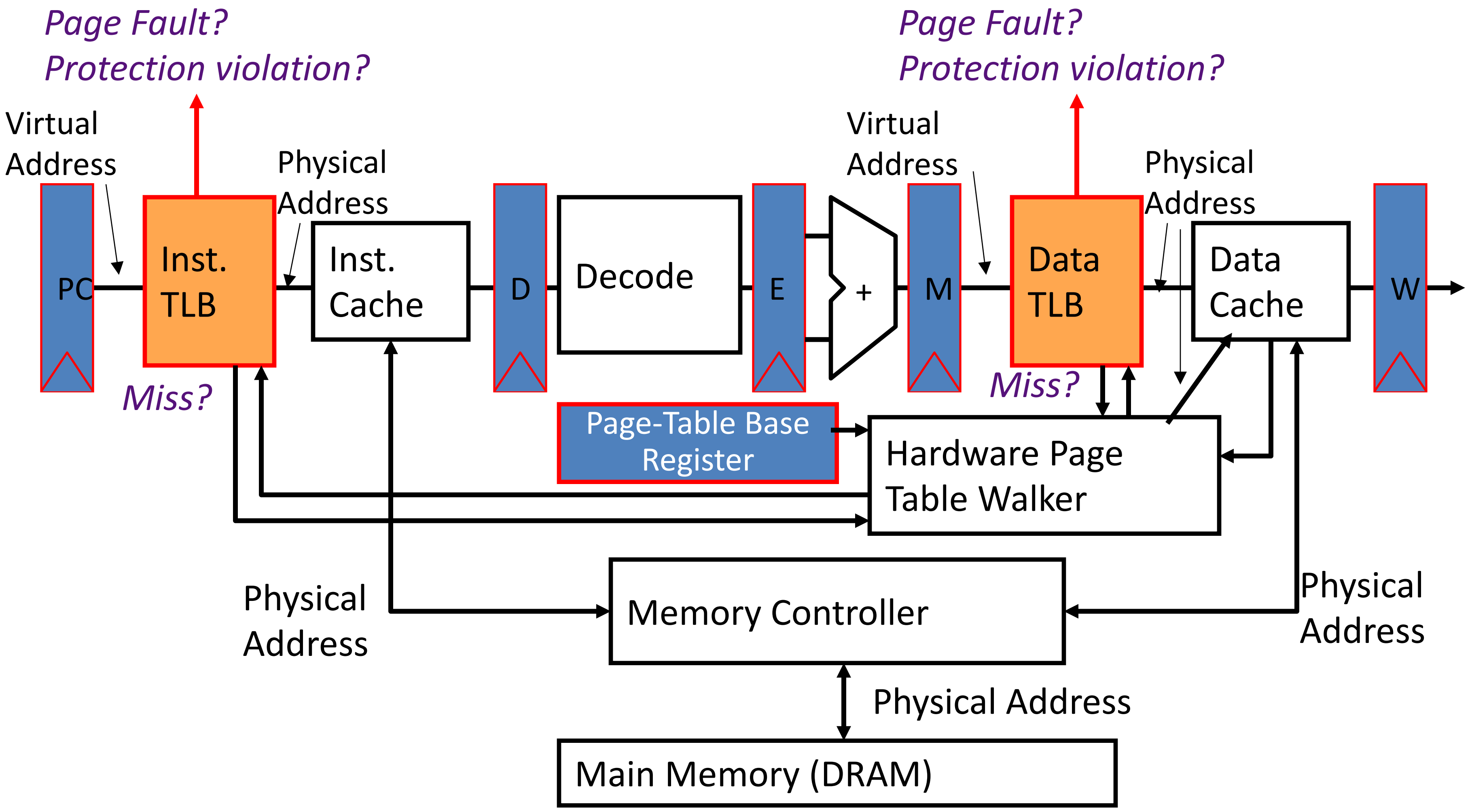
VM-related events in pipeline


PC
Inst TLB
Inst. Cache
D
Decode
E
+
M
Data TLB
Data Cache
W
TLB miss? Page Fault? Protection violation?
TLB miss? Page Fault? Protection violation?

- Handling a TLB miss needs a hardware or software mechanism to refill TLB
  – usually done in hardware now
- Handling a page fault (e.g., page is on disk) needs a *precise* trap so software handler can easily resume after retrieving page
- Handling protection violation may abort process

Page-Based Virtual-Memory Machine

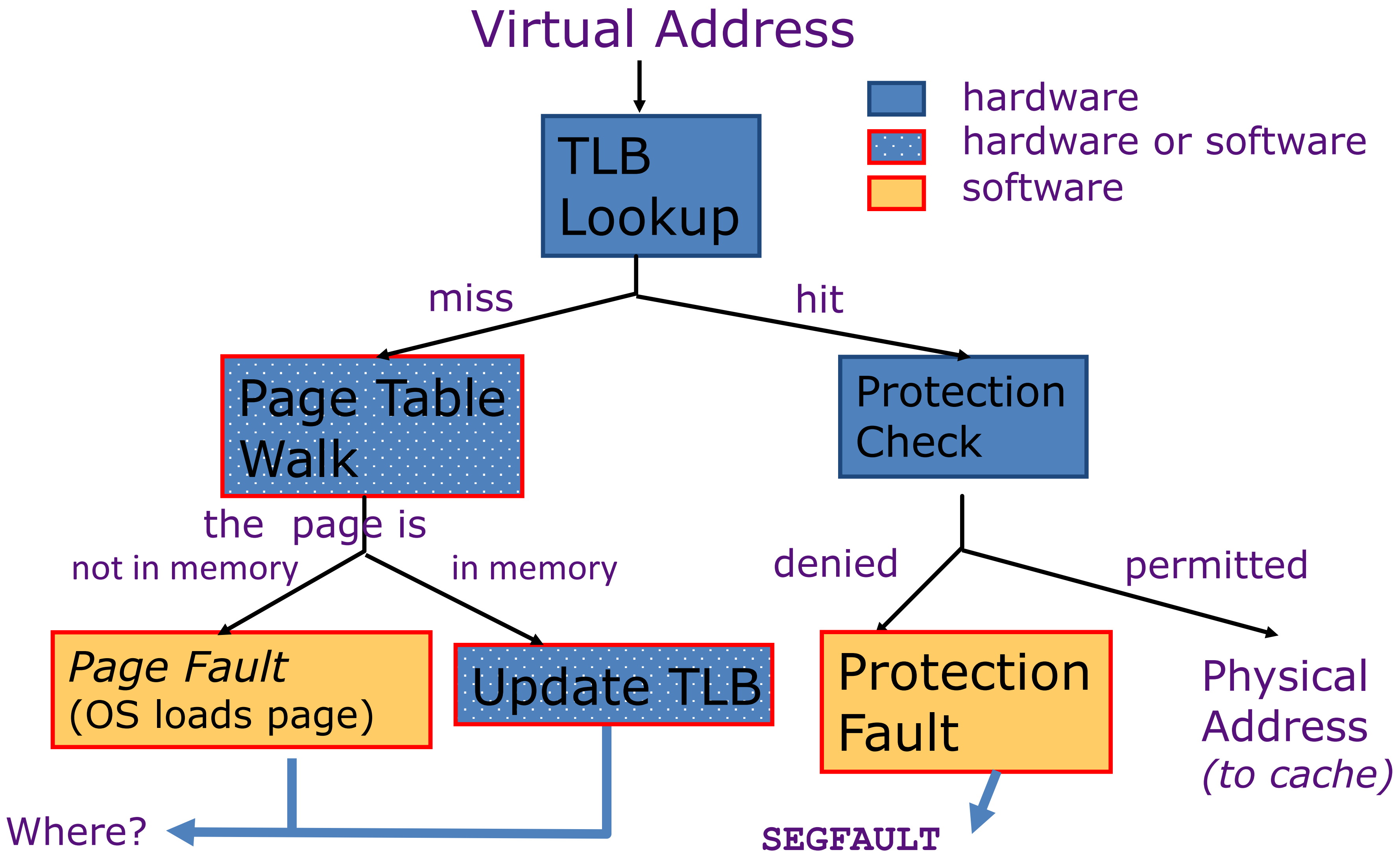
(Hardware Page-Table Walk)



- Assumes page tables held in untranslated physical memory

Address Translation:

putting it all together



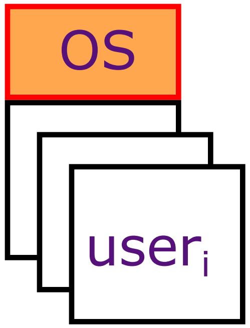
Modern Virtual Memory Systems

Illusion of a large, private, uniform store

Protection & Privacy

several users, each with their private address space and one or more shared address spaces

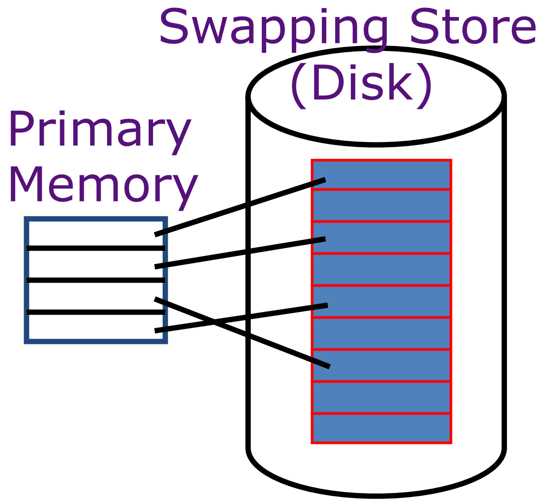
page table ≡ name space



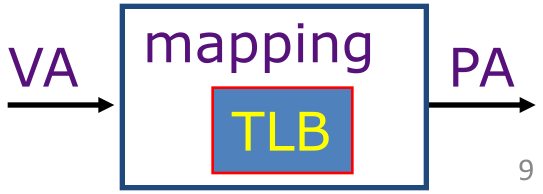
Demand Paging

Provides the ability to run programs larger than the primary memory

Hides differences in machine configurations

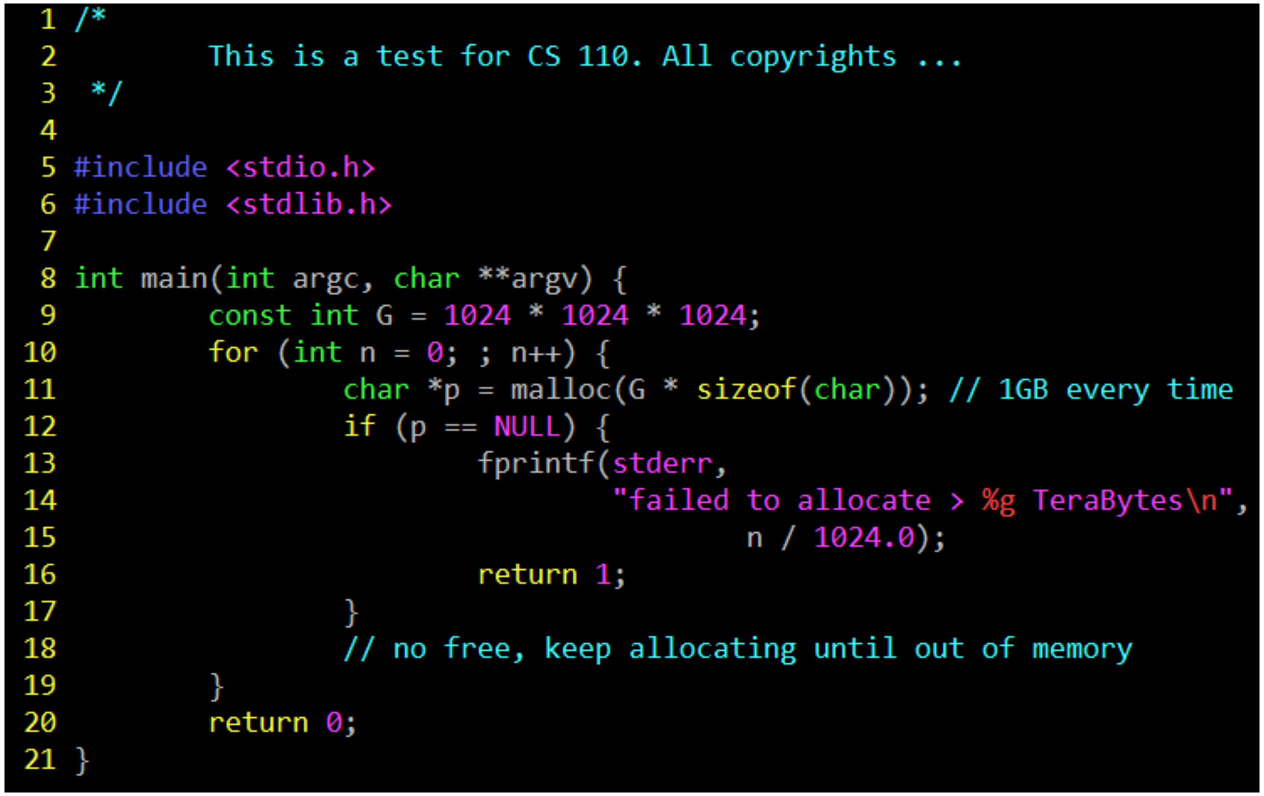


The price is address translation on each memory reference



Remember: Out of Memory

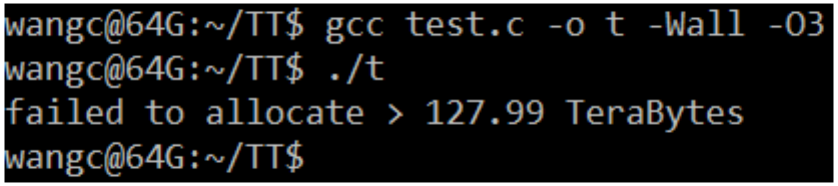
- Insufficient free memory: **`malloc()`** returns **`NULL`**



```c
/*
        This is a test for CS 110. All copyrights ...
 */

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
        const int G = 1024 * 1024 * 1024;
        for (int n = 0; ; n++) {
                char *p = malloc(G * sizeof(char)); // 1GB every time
                if (p == NULL) {
                        fprintf(stderr,
                                "failed to allocate > %g TeraBytes\n",
                                        n / 1024.0);
                        return 1;
                }
                // no free, keep allocating until out of memory
        }
        return 0;
}
```



```
wangc@64G:~/TT$ gcc test.c -o t -Wall -O3
wangc@64G:~/TT$ ./t
failed to allocate > 127.99 TeraBytes
wangc@64G:~/TT$
```

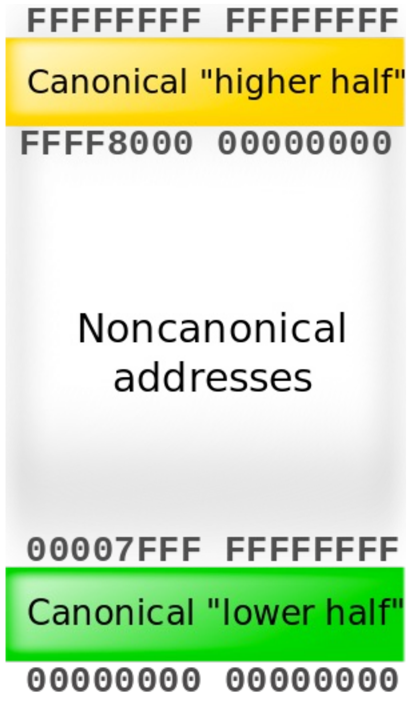
Limited VM Space with x86-64

- 64-bit Linux allows up to **128TB** of virtual address space for individual processes, and can address approximately 64 TB of physical memory, subject to processor and system limitations.
- For Windows 64-bit versions, both 32- and 64-bit applications, if not linked with "*large address aware*", are limited to **2GB** of virtual address space; otherwise, **128TB** for Windows 8.1 and Windows Server 2012 R2 or later.

Source: https://en.wikipedia.org/wiki/X86-64

48bit for address translation only

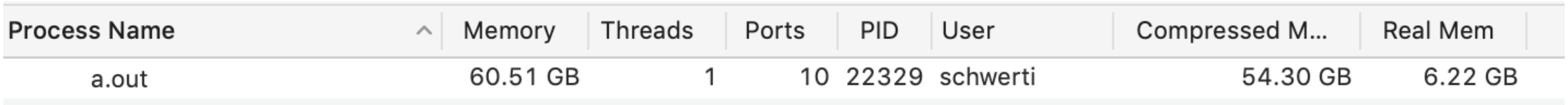
- Still provides plenty of space!
- Higher bits "sign extended": "canonical form"
- Convention: "Higher half" for the Operating System
- Intel has plans ("whitepaper") for 56 bit translation – no hardware yet



- https://en.wikipedia.org/wiki/X86-64#Virtual_address_space_details

Using 128TB of Memory!?

- A lazy allocation of virtual memory
  - Not used → not allocated
  - Try reading and writing from those pointers: works!
  - Even writing Gigabaytes of memory: works!
- Memory Compression!
  - Take not-recently used pages, compress them => free the physical page
- https://www.lifewire.com/understanding-compressed-memory-os-x-2260327



Process Name	Memory	Threads	Ports	PID	User	Compressed M...	Real Mem
a.out	60.51 GB	1	10	22329	schwerti	54.30 GB	6.22 GB

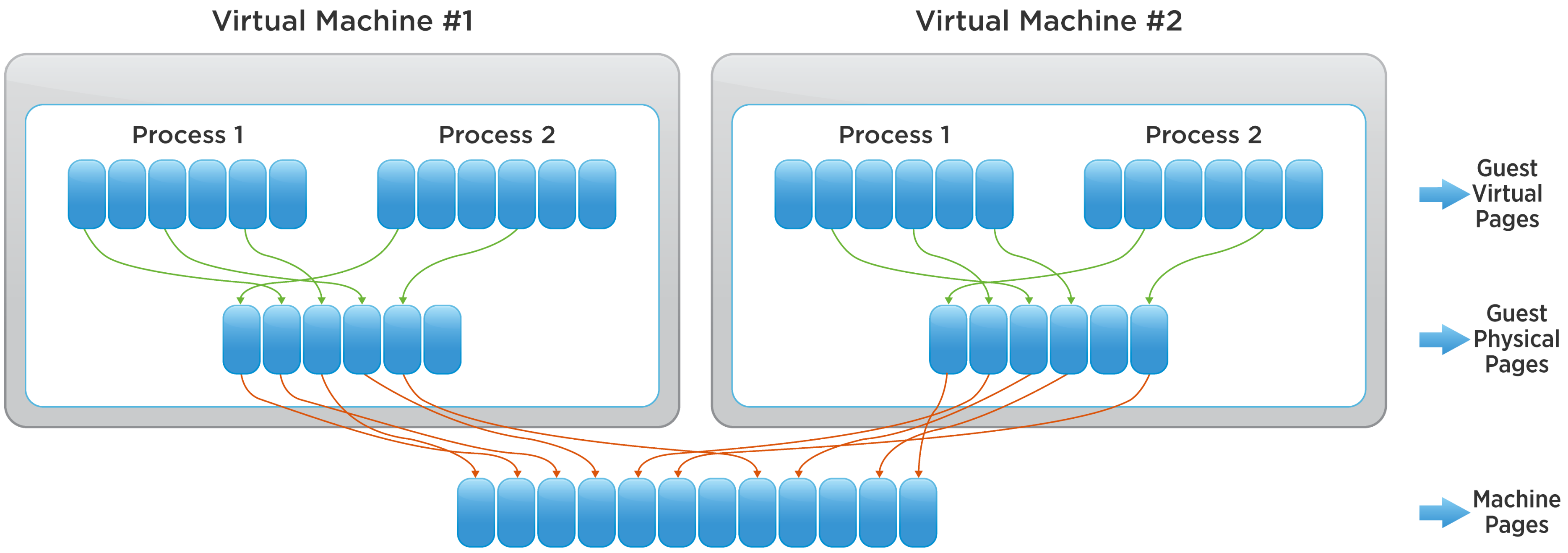
Virtual Machines

Virtual Machine

- Virtual Memory (VM) != Virtual Machine (VM)
  - Emulation: Run a complete virtual CPU & Memory & … - a complete virtual machine in software (e.g. QEMU)
  - Virtual Machine: Run as many instructions as possible directly on CPU, only simulate some parts of the machine) (e.g. VirtualBox)
- Last lecture: Supervisor Mode & Use Mode;
  now also: Virtual Machine Mode
  - Host OS activates virtual execution mode for guest OS =>
  - Guest OS thinks it runs in supervisor mode, but in fact it doesn't have access to physical memory! (among other limitations)
- CPUs support it (AMD-V, Intel VT-x), e.g. new Intel instructions: VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, and VMXON

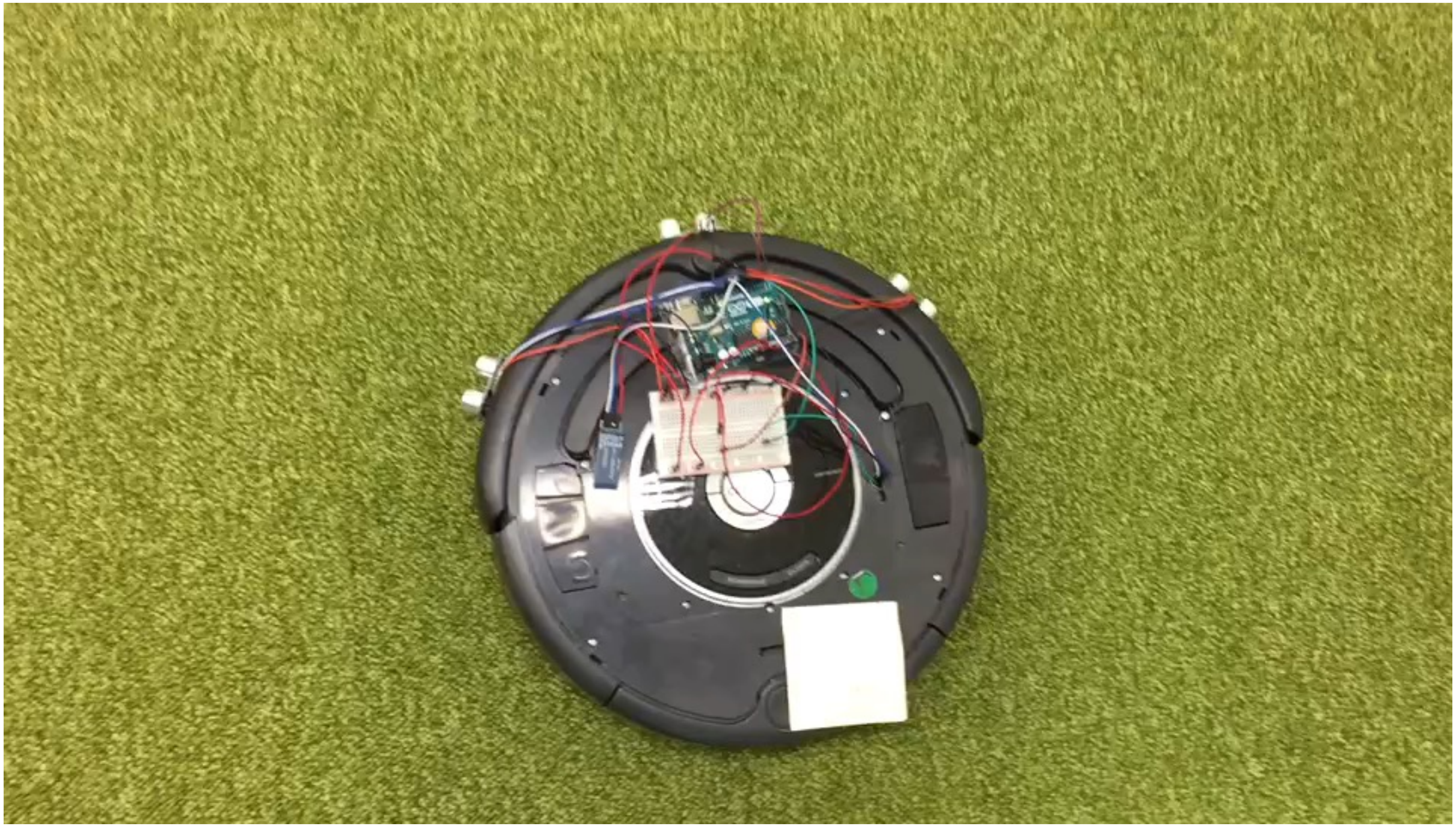
What about the memory in Virtual Machines?

- Need to translate Guest Virtual Address to Guest Physical address to Machine (Host) Physical address: Earlier the Guest part was done (transparently) in software by the Virtual Machine … now in hardware!



Embedded System Design

An example of embedded system



Embedded System

- An embedded system is nearly any computing system (other than a general-purpose computer) with the following characteristics
  - Specifically-functioned
    - Typically, is designed to perform predefined function
  - Tightly constrained
    - Tuned for low cost
    - Single-to-fewer components based
    - Performs functions fast enough
    - Consumes minimum power
  - Reactive and real-time
    - Must continually monitor the desired environment and react to changes
  - Hardware and software co-existence

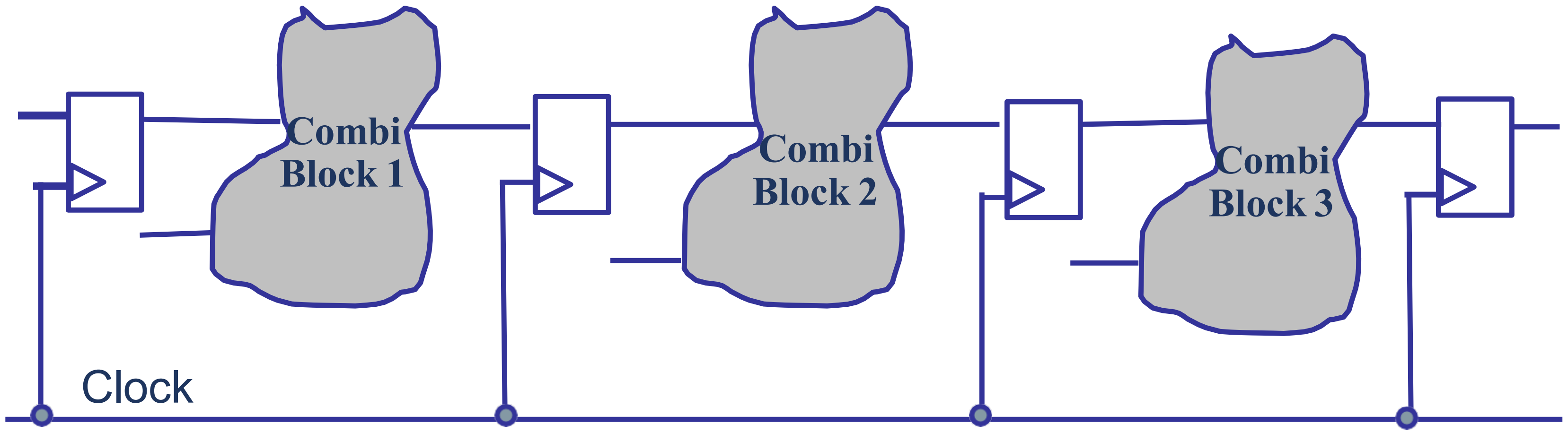
Embedded Systems Examples

- Examples:
  - Communication devices
    - Wired and wireless routers and switches
  - Automotive applications
    - Braking systems, traction control, airbag release systems, and cruise-control applications
  - Aerospace applications
    - Flight-control systems, engine controllers, auto-pilots and passenger in-flight entertainment systems
  - Defence systems
    - Radar systems, fighter aircraft flight-control systems, radio systems, and missile guidance systems

Major Design Metrics to be Considered

- Timing performance
- Power consumption
- Chip area (Cost)
- Technology
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- Time-to-market
- … …

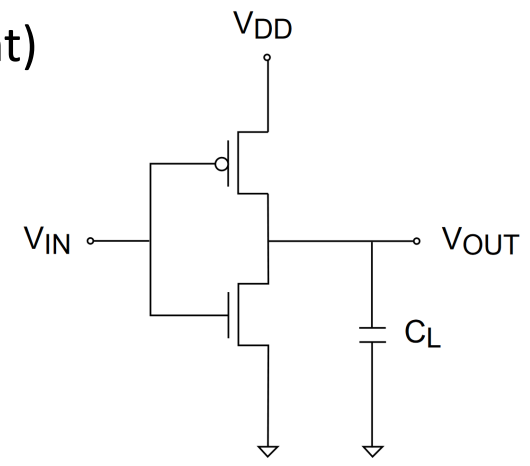
Timing Performance



- Clocks are used to synchronize the start of computations in all combinational blocks.
- Clock period is determined by finding out the **longest** path delay of all combinational blocks.
- All combinational blocks are supposed to finish the computations of the current clock cycle before the start of the next clock cycle.

Power Consumption

- Why low power?
  - High performance and integrity of VLSI circuits
  - Popularity of portable devices
- Power consumption in CMOS circuits
  - Dynamic power dissipation (used to be dominant)
    - Charging and discharging capacitors
  - Short-circuit power dissipation
  - Leakage power dissipation (increasingly larger)
- Dynamic power dissipation



$$P_{dynamic} = \alpha \cdot C_{phy} \cdot V_{dd}^2 \cdot f_{clk}$$

where α: switching activity, C_{phy}: physical capacitance,

V_{dd}: supply voltage, f_{clk}: clock frequency

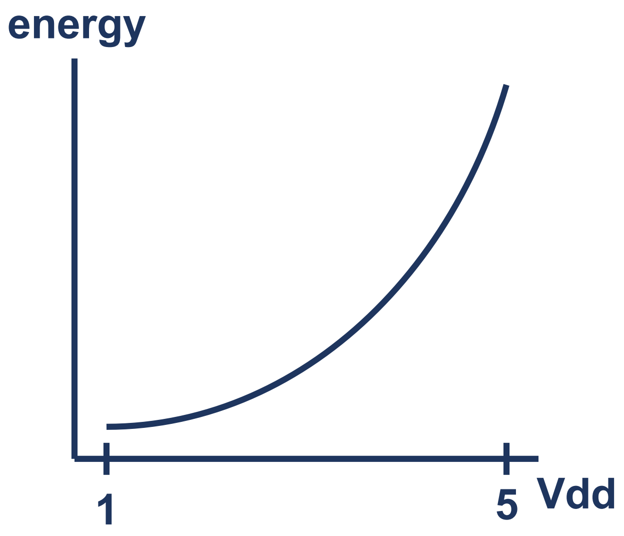
Power Consumption

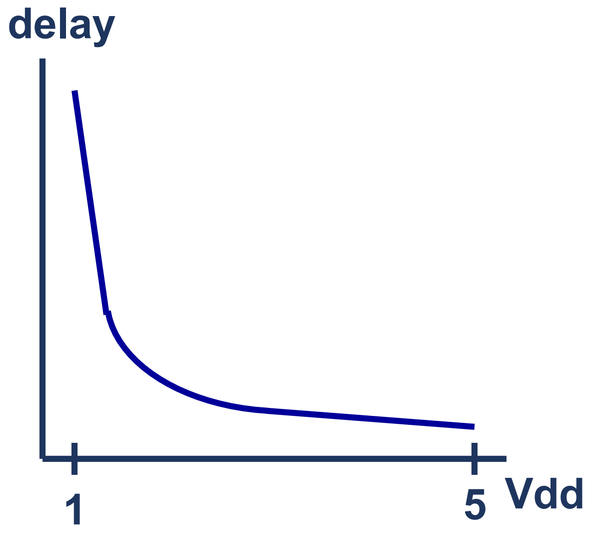
- Supply voltage reduction
  - Quadratic effect of voltage scaling on power

$$P_{dynamic} = \alpha \cdot C_{phy} \cdot V_{dd}^2 \cdot f_{clk}$$

5V → 3.3V ➔ 60% power reduction

  - Supply voltage reduction ➔ increased latency

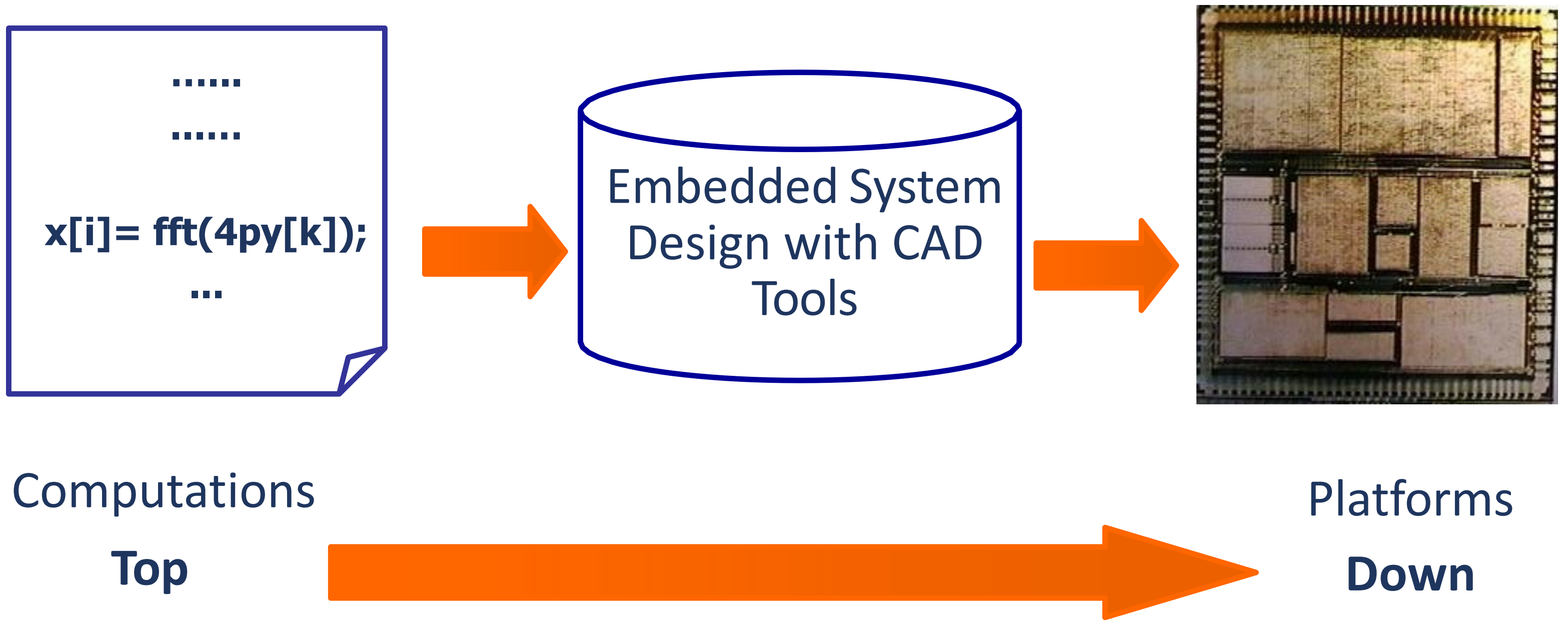




Who Contributes to Embedded System Designs

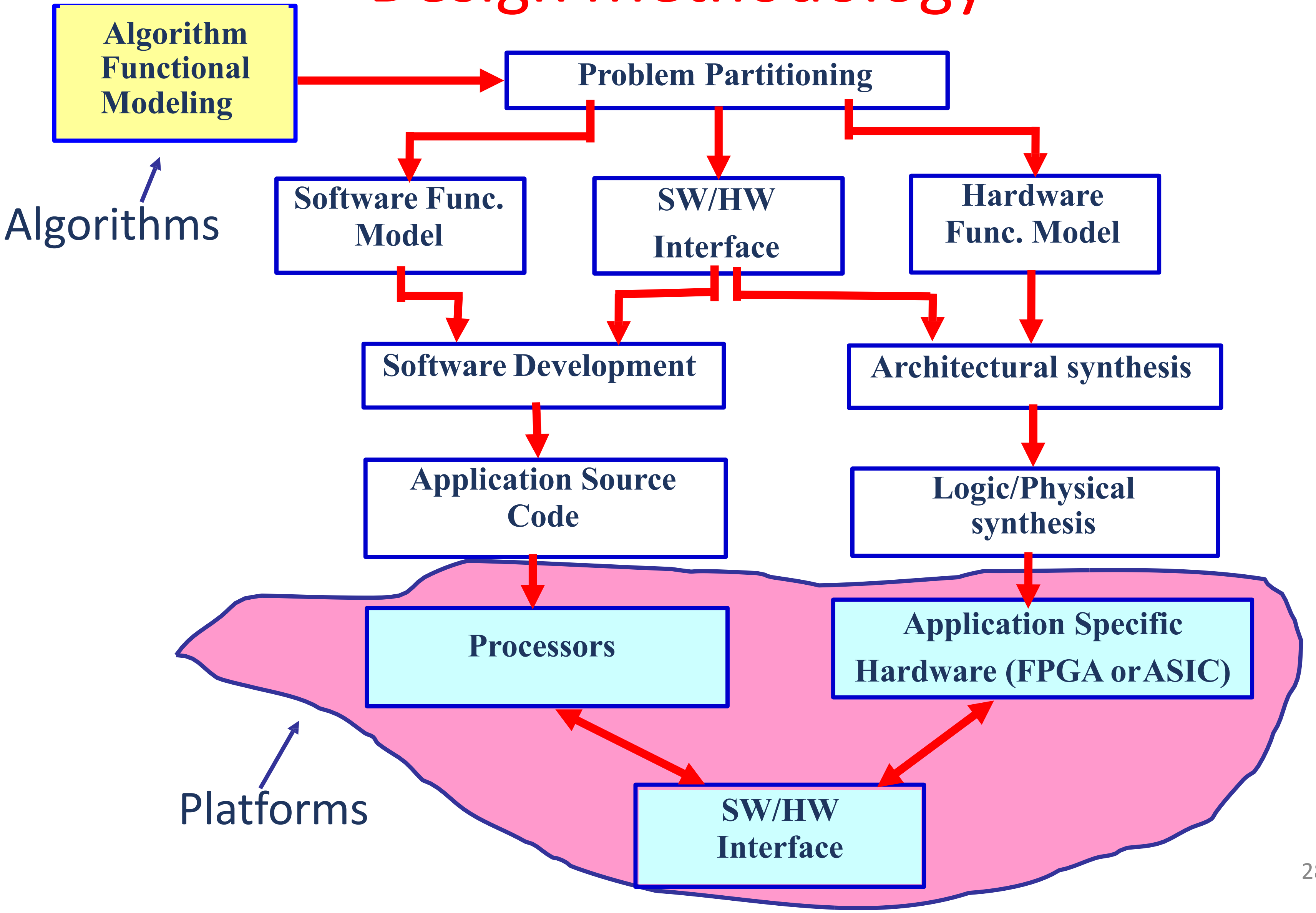
- Application algorithm developer
  - e.g., telecommunication, multi-media researcher
- Computer-Aided Design (CAD) tool developer
  - e.g., Synopsys, Cadence companies
  - Potential research field
- IC designers working at different levels
  - e.g., IC design group in Infineon, Broadcom and HP
  - Industry field
- Test engineer
  - Both research and Industry field

Embedded Systems Implement Computations on Platforms



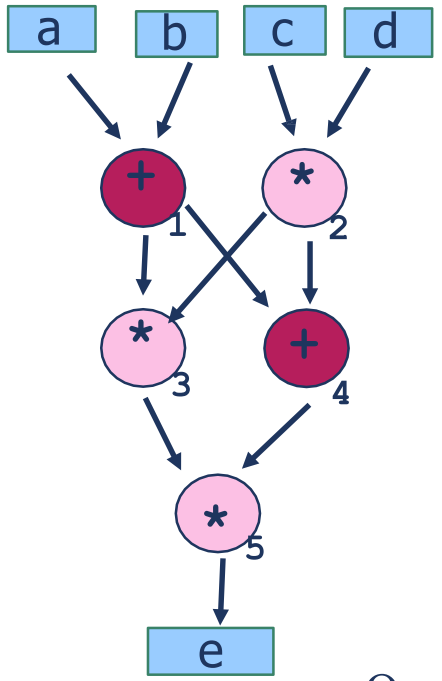
We will examine the ***design methodologies*** to implement computations (algorithms) on platforms. We temporarily forget analog design for a moment.

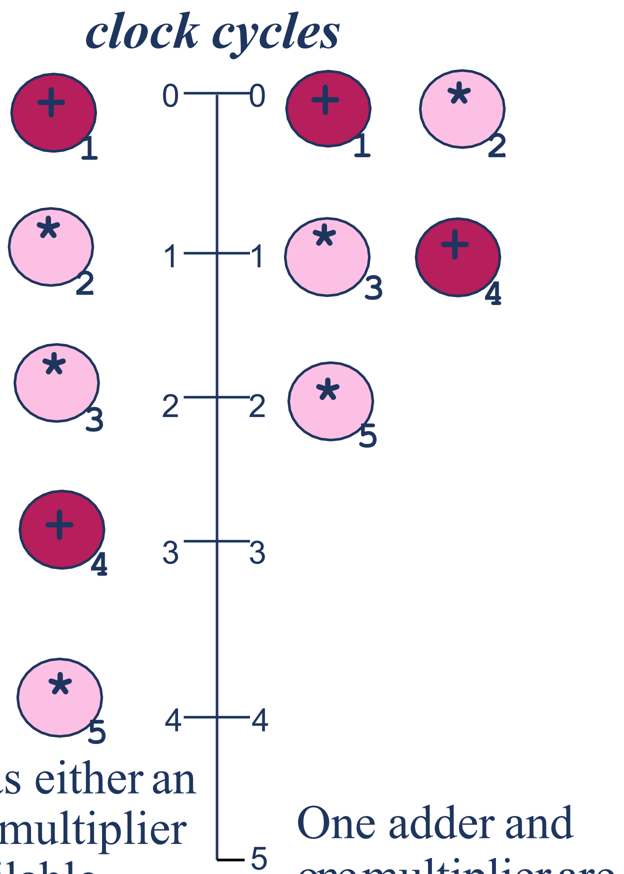
Simplified and General Embedded System Design Methodology



An Example to Start

```
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3= tmp0 + tmp1;
e = tmp2 * tmp3;
```



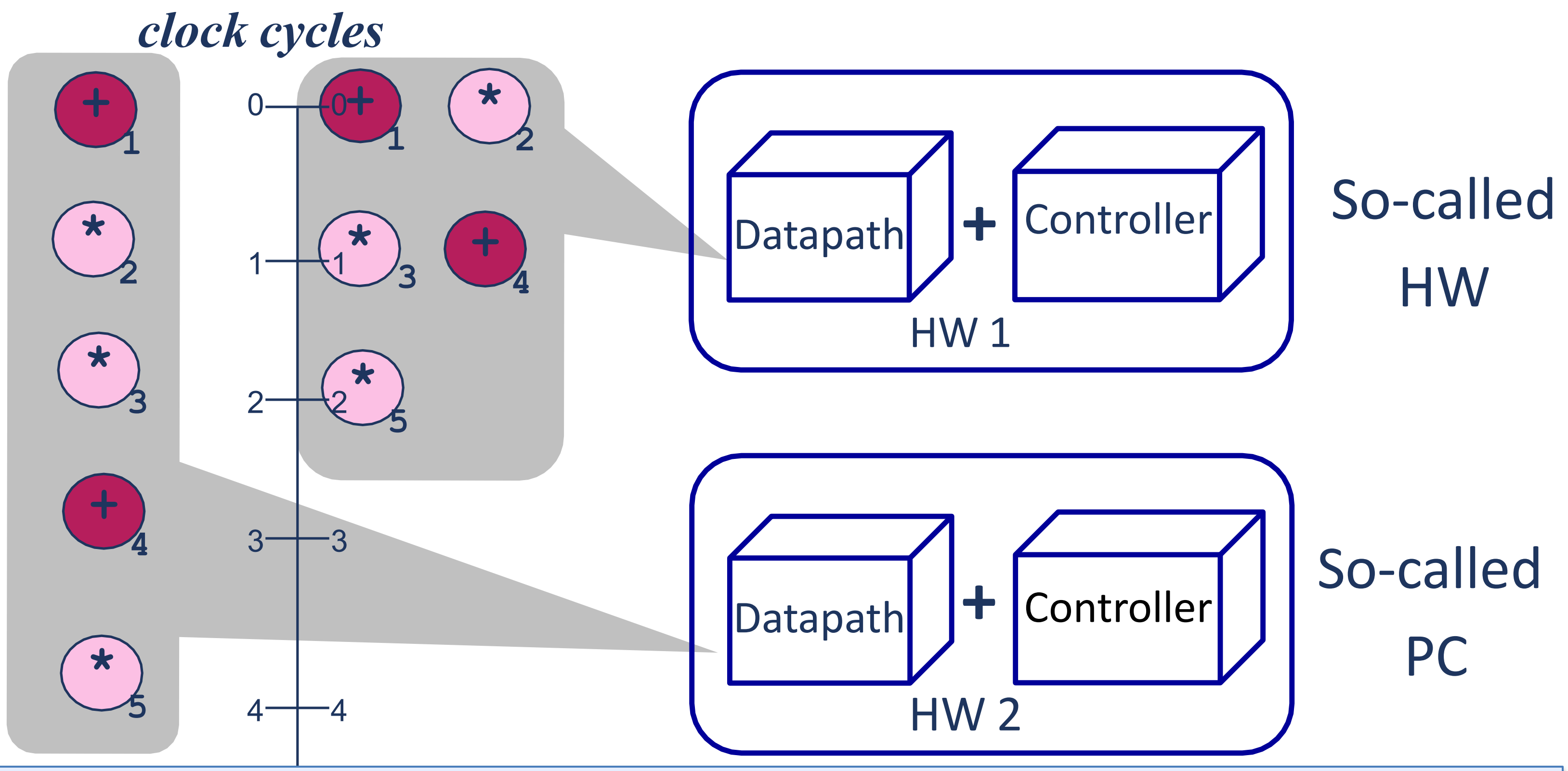


One ALU as either an adder or a multiplier is available

One adder and one multiplier are available

Algorithm Code   Data Flow Graph   schedule 1   schedule 2

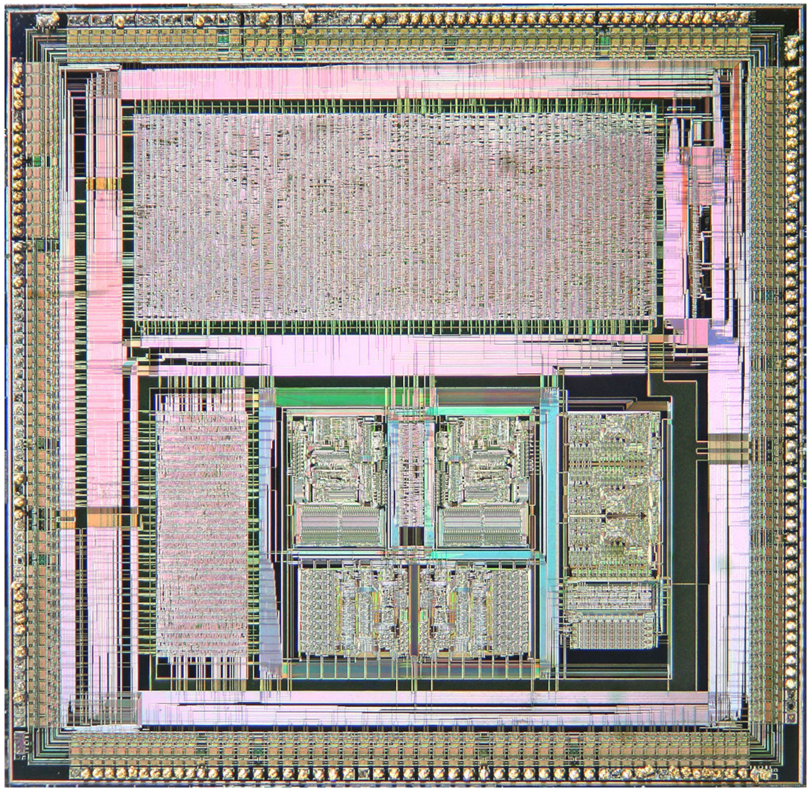
Datapath and Controller



- Datapath implements operators, decides the area and speed that a design can achieve.
- Controller decides which operator of a datapath should work at specific cycle according to schedules.
- Embedded system design is actually to design the datapath and controller.

ASIC

- Application-specific integrated circuits



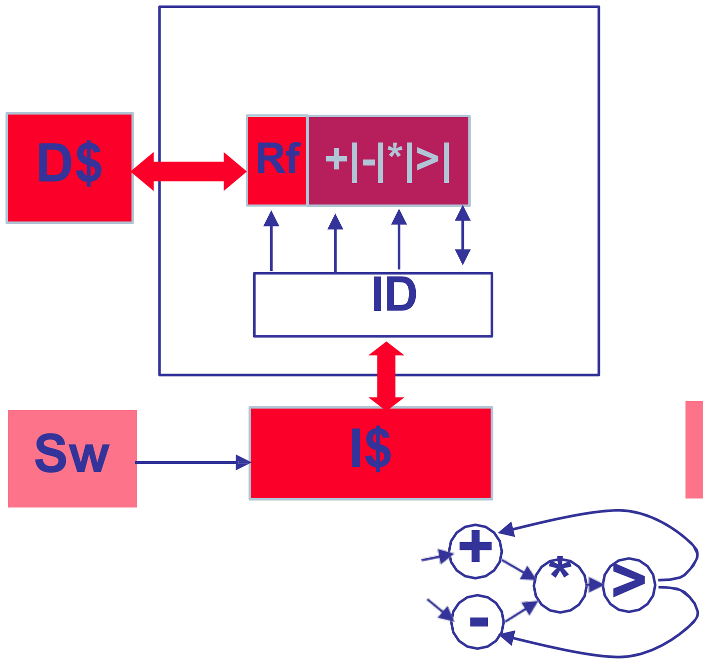
A custom ASIC (486 chipset) showing gate-based design on top and custom circuitry on bottom (Source: Wikipedia)

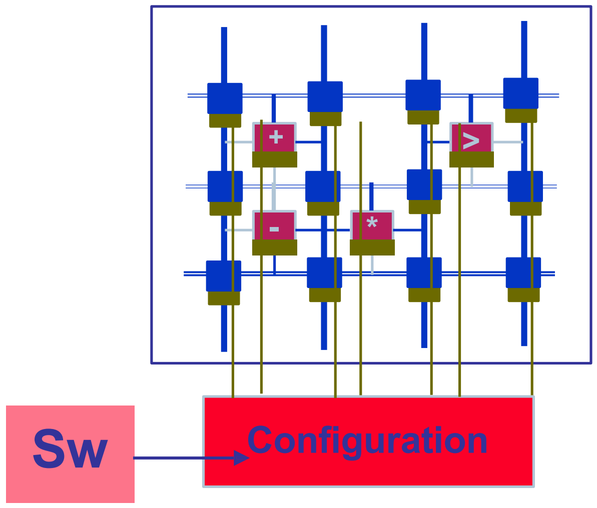
ASIC

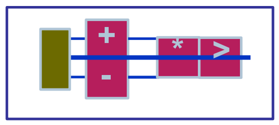
- Application-specific integrated circuits
- De Morgan’s theorem
  - Theoretically we only need 2-input NAND or NOR gates to build anything
- ASIC is good, but
  - High risky and expensive to design and manufacture
    - Suitable for very high-volume mass production
  - Permanent circuitry
    - Once designed, not changeable

Three Kinds of Embedded System Implementation Choices

Processor	Reconfigurable FPGA	ASIC







Processor	Reconfigurable FPGA	ASIC
Programmable	Configurable	No wiring
Sequential	Parallel wired algorithm	No configuration overhead
Instruction flow (cycle)	"Program" flow (occasionally)	
Transfer bottleneck	Distributed data	
Power: 100	**10**	**1**

Why Use Reconfigurable Hardware?

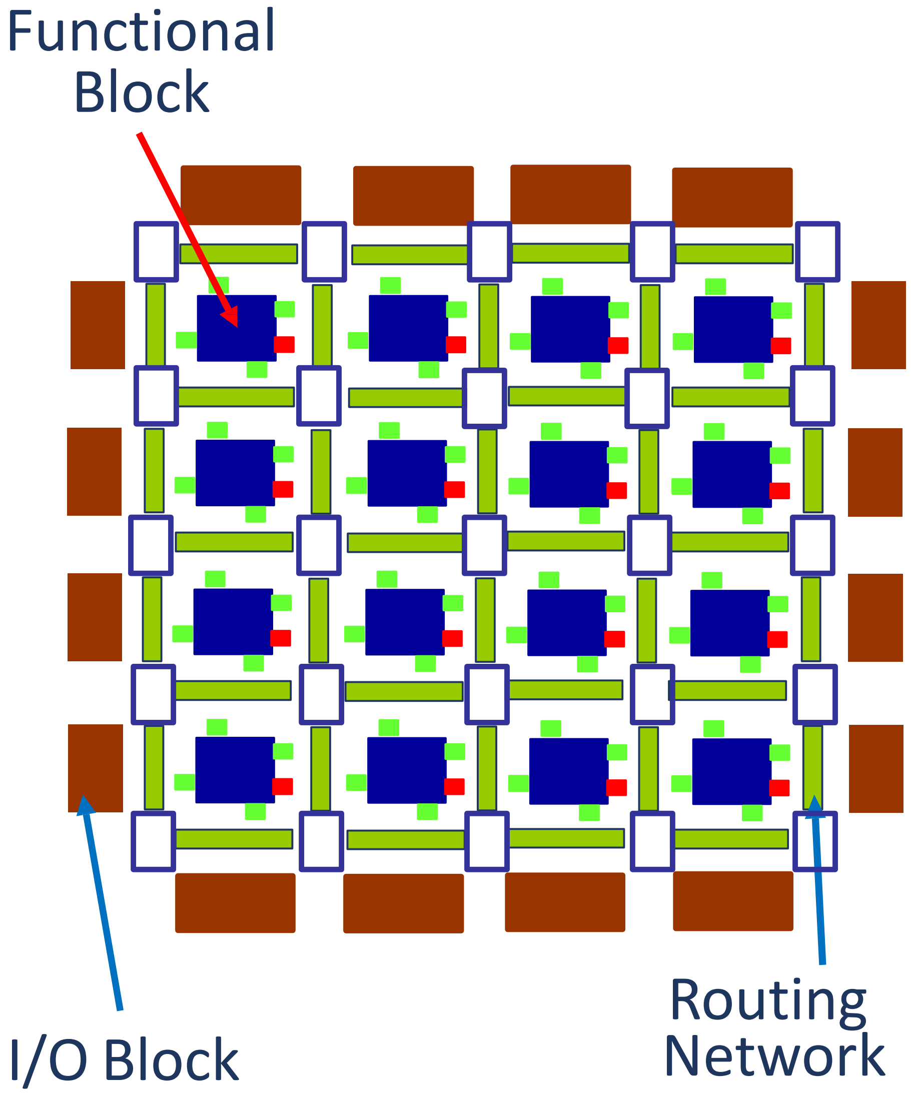
	Processor	FPGA	ASIC
Performance	Low	Medium	High
Flexibility	High	High	Low
Power	High	Medium	Low

Why FPGAs?

- Combine flexibility with performance.
- Shorter time-to-market and longer time-in-market.

Architecture of FPGA

Simplified FPGA Architecture

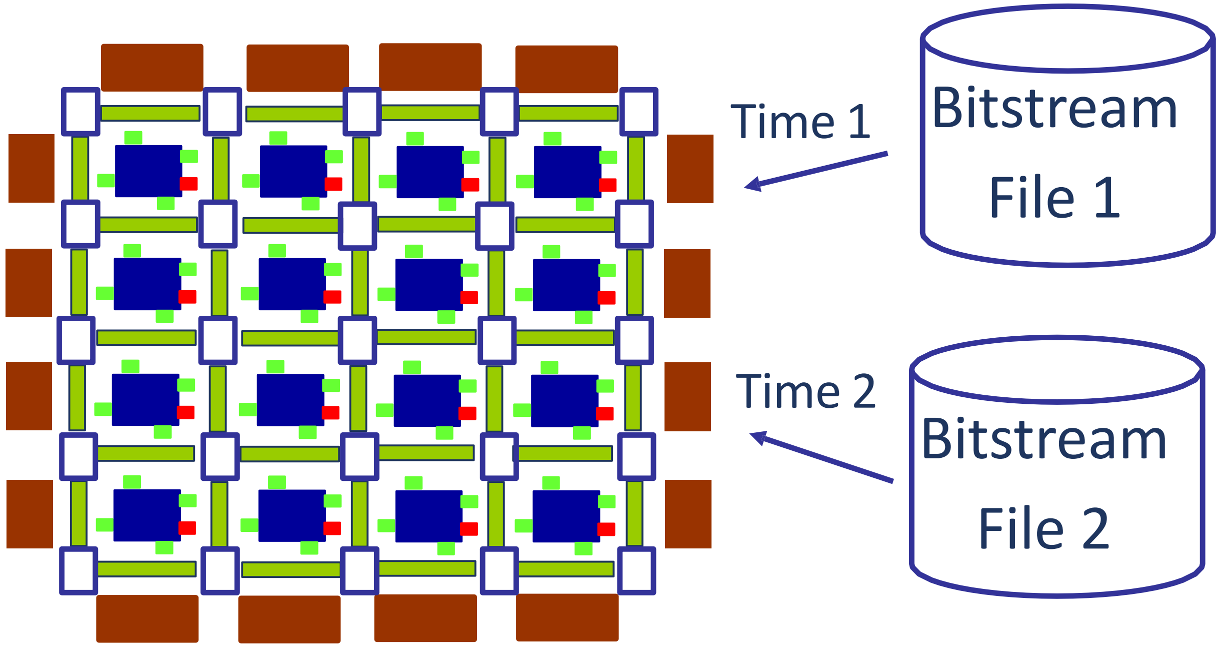


All the three FPGA components can be re-programmed with configurations to implement application-specific digital circuits.

For example,

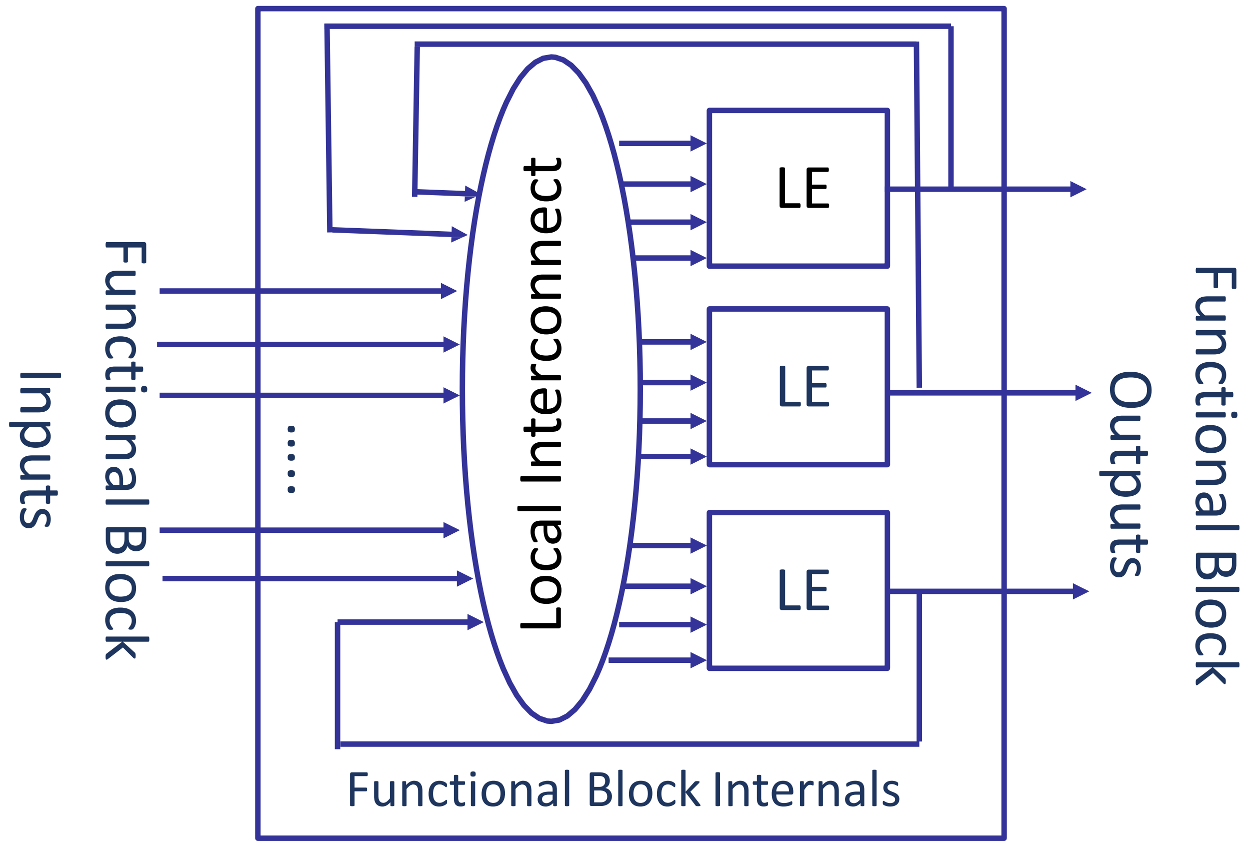
- each functional block can be programmed to implement a small amount of digital logic of a design;
- the routing network can be programmed to implement the design specific interconnection pattern;
- I/O blocks can be programmed to implement the input and output ports according to design requirements.

FPGA Reconfiguration



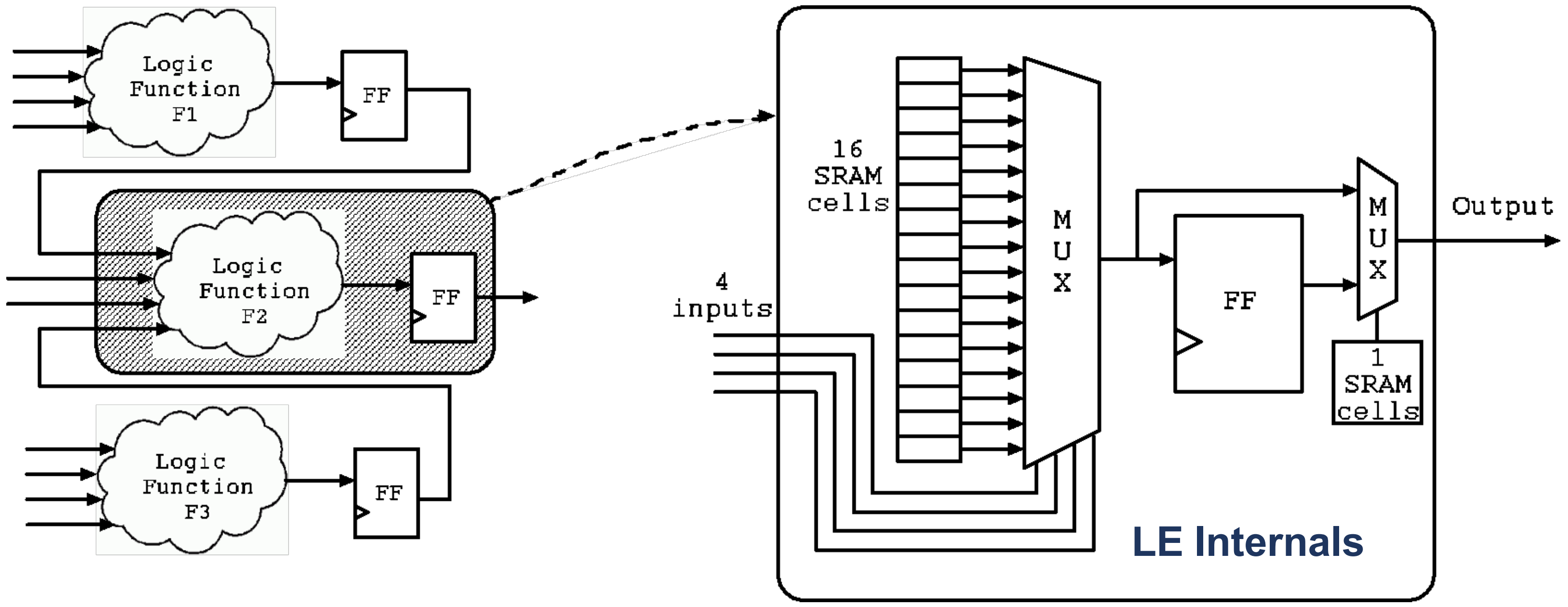
- All the programming information for the three FPGA components is stored in a configuration file. The configuration file for a FPGA is often called a *bitstream* compared to a binary executable for a processor.
- Once a bitstream for a digital logic design is downloaded to a FPGA, the FPGA is programmed to implement the design.
- By providing different bitstreams, a single FPGA can be re-programmed to implement different designs at different times.

FPGA Functional Block



- FPGAs use the Look-Up Table (LUT) type of functional block.
- A functional block is normally made of one or several logic elements (LE).
- Functional blocks differentiate from each other mainly in terms of the input size of an LE and the number of LEs in a functional block.
- State-of-the-art FPGAs normally use 4-input LEs.

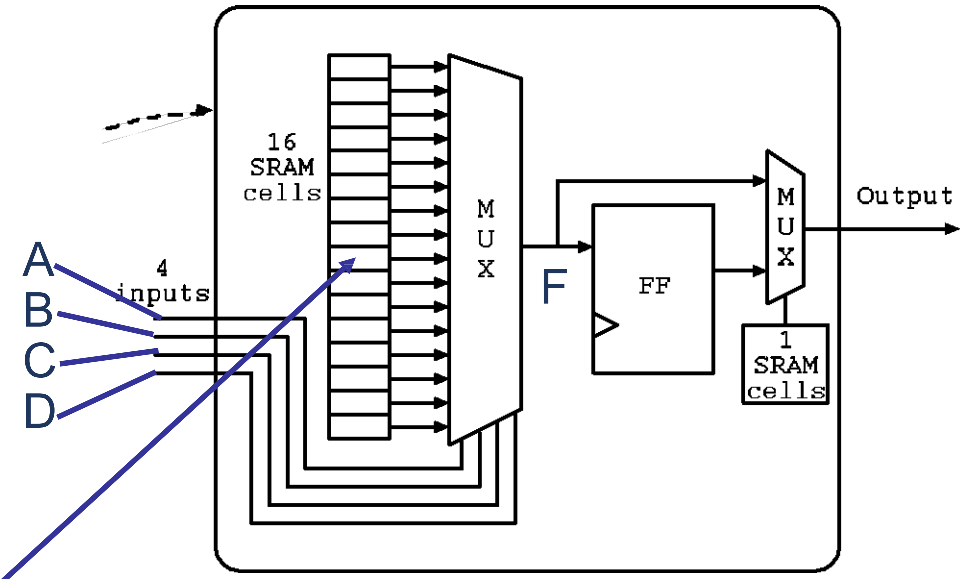
FPGA Logic Element



- One LE consists of a 16 SRAM cell **Look-Up Table (LUT),** and a flip flop (FF).
- The 16 SRAM cells LUT stores the truth table of any 4-input logic function. Thus it can implement any 4-input logic function.
- The FF implements the storage element in a sequential circuit.

LUT Content

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

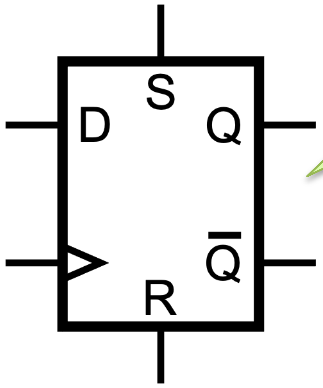


$$F = (A \,\&\, B) \mid (C \,\&\, D)$$

- The 16 SRAM cell LUT stores the output column of the truth table of the F function.
- The 4 inputs A, B, C and D will determine which bit the F value is for the current values of A, B, C and D.

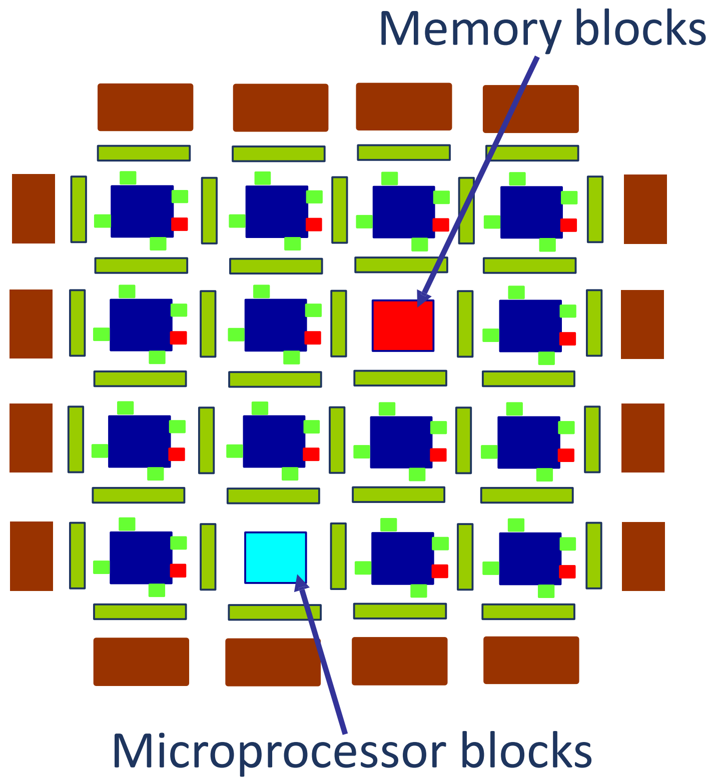
FF

- D Flip Flop
  - A data storage element
  - When FF sees a rising edge of the clock, it registers the data from the Input D to the Output Q.



How long can a D Flip Flop hold current data?

Additional Computational Resources

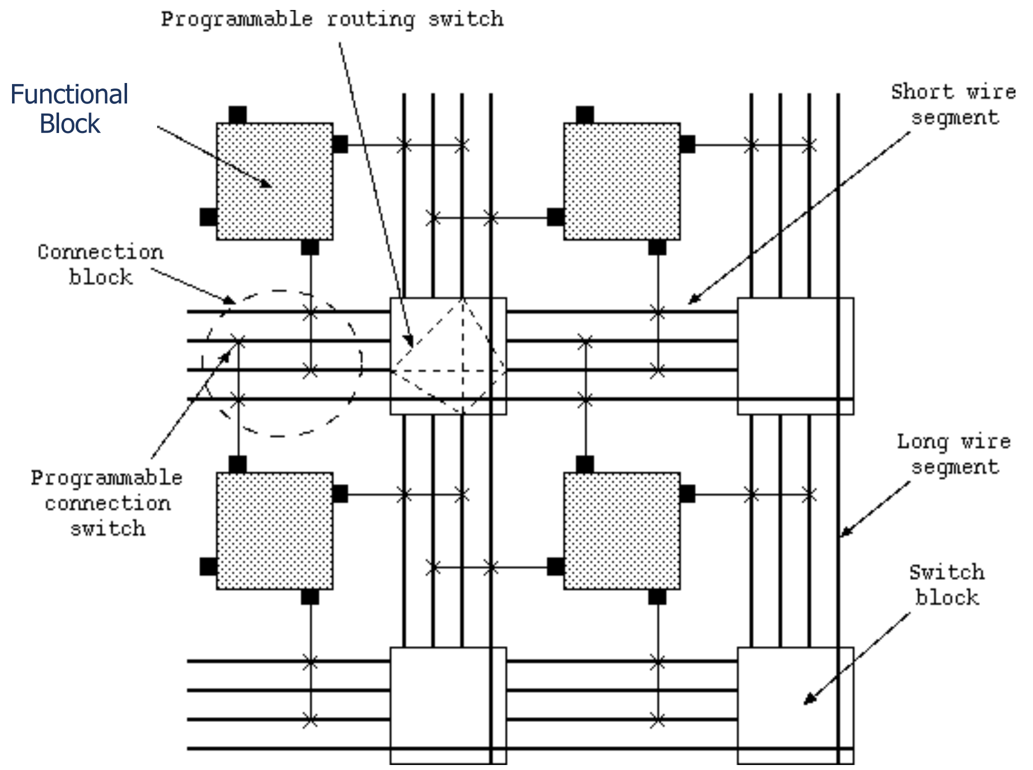


- Besides the LEs, some functional blocks in different FPGAs have architecture specific features to improve the performance when implementing arithmetic functions.
- These architecture specific features include carry logic, embedded memory blocks, multiplier and other hard cores.
- Hard cores generally implement functions efficiently compared to FPGA functional blocks.

Routing

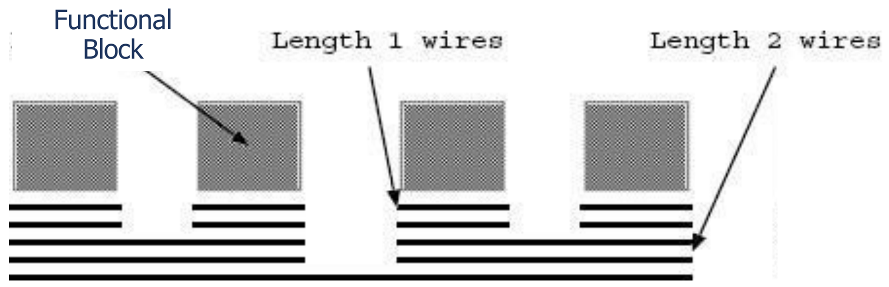
- Routing means interconnecting
  - Through programmable wires and switches
  - Between functional blocks, and between I/O blocks and functional blocks
- Routing is a challenging problem
  - Routing technique used in an FPGA largely decides the amount of area used by wire segments and programmable switches, as compared to area consumed by functional blocks.
  - Inferior routing may lead to congestion or failure of signals.

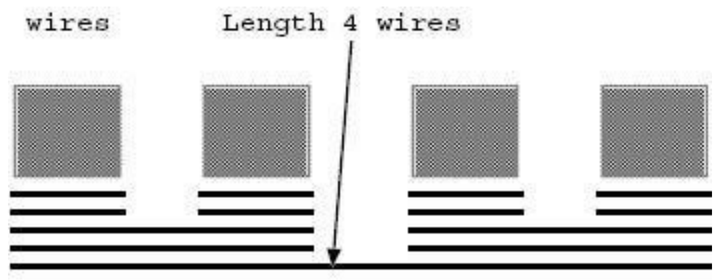
FPGA Routing Architecture



- A functional block input or output pin can connect to some or all of the wiring segments in the channel adjacent to it via a *connection block* of programmable switches.
- At every intersection of a horizontal channel and a vertical channel, there is a *switch block*. It is a set of programmable switches that allow some of the wire segments incident to the switch block to be connected to others.
- By turning on the appropriate switches, short wire segments can be connected together to form longer connections.

FPGA Routing Wires





- Some FPGAs contain routing architectures that include different lengths of wires.
- The length of a wire is the number of functional blocks it spans.
- Left figures show wires of length 1, 2 and 4.
- Long wires introduce shorter delays for long interconnections since fewer switch blocks will be passed.

Conclusion

- TLB
  - Accelerate address translation
- Embedded system
- ASIC, processor
- FPGA
  - Functional block, routing, etc.